← Back to blog

TECHNOLOGY & INNOVATION

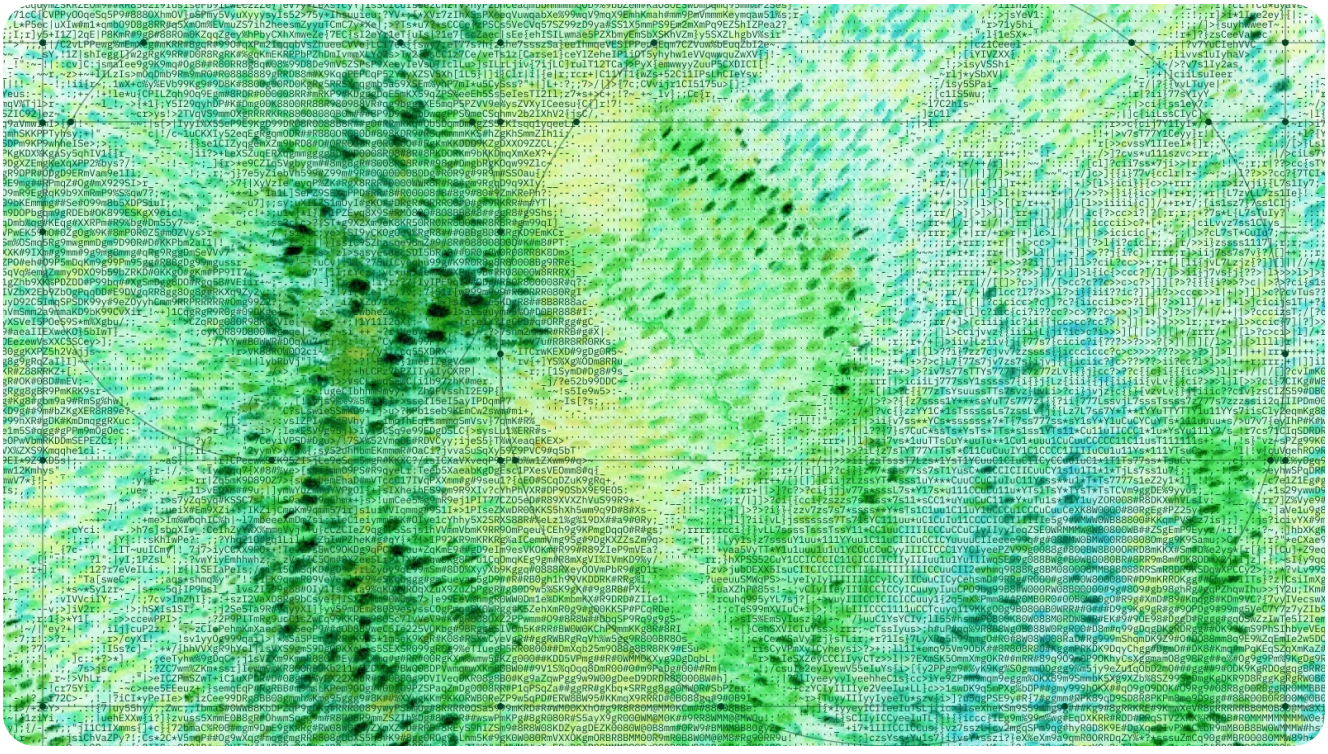
From spreadsheets to durable systems: how Headway rebuilt credentialing with Temporal

At Headway, credentialing providers quickly and accurately is one of the most important ways we expand access to care. We recently rebuilt that system from the ground up, moving from spreadsheets to a durable platform powered by Temporal that now supports more than a dozen teams and tens of thousands of providers.

SL

Stephen Leicht

Engineer · Apr 22, 2026



Every mental health care provider (therapist, psychologist, psychiatrist, etc.) who joins Headway goes through credentialing: a background verification process that confirms their licenses, checks them against third-party databases, handles state-specific requirements, and ultimately clears them to bill insurance plans for patient visits. It helps us onboard providers who meet the quality standards of Headway and each insurance company they'll work with.

It's the kind of process that sounds straightforward until you look at what's actually involved: external APIs that take days to respond, state-by-state regulatory checks that can run in parallel or depend on each other, and decision points where a human at Headway needs to review a case before the system moves forward.

In the early days of Headway, this entire process ran on spreadsheets. Contractors manually entered each provider into third-party systems, ran spreadsheet formulas to validate the results, downloaded CSVs, uploaded them to other tools, downloaded more CSVs from those tools, and finally uploaded them back into our system. It was a web held together by spreadsheet logic, manual processes, and institutional knowledge. That's the kind of scrappy, "first time is hand made" approach that gets a startup off the ground. But once you're at the scale of onboarding hundreds of providers every day, spreadsheets and manual processes don't scale with you. That's why, in spring 2025, we decided it was time to automate this process. The smoother and easier this process could be meant we could onboard more providers which meant more access to care, our ultimate mission at Headway.

What needed to change

Up to this point we had Celery (a simple task queue library for python) for async tasks. It's still part of our stack and it works fine for what it's good at: short-lived, fire-and-forget async tasks. We still have many periodic jobs running on Celery today and they do their thing reliably.

But the credentialing process isn't a short-lived task. It's something that can run for weeks (sometimes months) with steps that depend on external systems updating on their own timeline, human decisions that can't be predicted, and branching logic that changes based on provider type, state, and edge cases we discover along the way. Our Celery implementation fell short in a few notable ways:

- **No durable persistence.** The way we have Celery set up gives no guarantee that the tasks will execute, for example tasks getting dropped between deploys. For a

process that takes weeks, that's a non-starter. We would have needed to build a resilient persistence layer on top of Celery.

- **Complex orchestration required a complex and proprietary syntax.** Celery's chaining mechanism works for simple A-then-B flows, but modeling the kind of branching and looping credentialing requires (where the path forward depends on results from previous steps, external system states, and human decisions) meant wrestling with abstractions that were hard to reason about and harder to debug.
- **No built-in retry semantics worth trusting.** For a high-stakes process where getting it wrong means a provider can't see patients, we needed retry logic that was predictable and configurable, not bolted on.

We considered building what we needed on top of Celery. Within our engineering team, we recognized we had a dual mandate for fixing this system: build new automated credentialing workflows, but do it in a way that all teams at Headway can easily solve these problems for their own use case. Investing in making Celery do things it wasn't designed to do didn't serve either goal. Enter Temporal, a more advanced orchestration platform that provides durable, robust, and complex logic flow out of the box, and does it with a pretty shallow learning curve.

It's just code

Temporal mapped really well to the real world processes we were implementing. Our team did a series of workflows which were broken into individual steps. This matched workflows and activities to a tee. Even better, Temporal creates a log of each workflow and the progress through activities so if it's dropped, crashes, a new version is deployed etc, it just picks back up where it left off. Exactly the same as the manual process we were modeling.

The thing that made Temporal click for us, almost immediately, was that workflows are just code. It's not a complex DSL (domain specific language) we needed to learn like Celery's chain API or YAML. It is just Python code with if-statements, for-loops, and function calls which allows you to easily model anything you want with the tools you already know.

The first pattern we had to model was calling an external API and then waiting, sometimes days, for their system to update before we could continue. In Temporal, this became a polling loop with what we call a "skippable sleep": the workflow sleeps for 12 hours, then checks the external system. If it's not ready, it sleeps again. The sleep is implemented as a signal wait with a timeout, which means during

development or debugging, you can send a signal from the UI to skip the wait and check immediately. Because of the durability guarantees Temporal gave us out of the box this was easy to model in code and easy to rely on over days if not weeks. The whole pattern is less than 10 lines of Python:

```
while not result:
    result = await workflow.execute_activity(
        check_external_system_status, ...
    )
    if not result:
        # Sleep 12 hours, but allow a signal to skip the wait.
        await workflow.wait_condition(
            lambda: self._skip_wait, timeout=timedelta(hours=12)
        )
        self._skip_wait = False
```

This ended up being very simple, and most importantly, debuggable. If the worker restarts mid-sleep, Temporal replays the workflow history and picks up right where it left off. No persistence layer to build, no state machine to maintain.

But the real "aha" moment came when we modeled multi-state credentialing, i.e. when a provider is getting credentialed in more than one state at a time. During this process, sometimes all states clear at the same time and the provider goes live everywhere at once. But sometimes one or more states have requirements that take longer (For example additional documentation requirements based on location or kind of license the provider has) and we don't want to hold up bringing that provider "live", and being able to treat patients in the states that are cleared.

In the spreadsheet era, operators handled this by manually splitting the provider into separate rows and tracking each state independently, which was messy to say the least. In Temporal, it's just parallel child workflows:

```
# Fan out one child workflow per state.
state_handles = []
for state in active_states:
    handle = await workflow.start_child_workflow(
        StateQualityCheckAndGoLiveWorkflow,
        StateCheckInput(provider_id=provider_id, state=state),
```

```
        )
        state_handles.append(handle)
    # States report back via signals as they complete.
    # Bring live the ones that are ready, keep monitoring the rest.
```

Each state runs its own checks independently, signals back to the parent when it's done, and the parent brings states live as they clear, no manual intervention required. Because it's just code, modeling this could be as nuanced as the messy real-world process it replaced.

When automation meets human judgment

Credentialing isn't fully automatable. There are steps where a person at Headway needs to make a judgement call. For example a provider who was previously deactivated could try to rejoin and needs manual approval, a third-party check flags something that requires document review or a committee needs to weigh in on an edge case. Standard approvals and obvious rejections are handled automatically, but these edge cases require us to do our due diligence and it's important that these decision points keep a human in the loop to ensure a high quality provider network.

The pattern we built for this is straightforward: when a workflow reaches a step that requires human input, it creates a task record in the database and enters a waiting state. That task shows up in our internal admin tool for an operator to review. They make their decision, click a button, and that action sends a Temporal signal back to the waiting workflow, which picks up and continues down the appropriate path. The workflow can wait up to 60 days for that signal, durably, with no compute cost while it waits.

This changed our Credentialing Operation team's day-to-day in a meaningful way. They went from doing everything manually (entering data into third-party systems, running spreadsheet validations, shuttling CSVs between tools) to focusing exclusively on the decisions that actually require human judgment. The tedious mechanical work was automated; what's left is the work that genuinely benefits from a person's expertise.

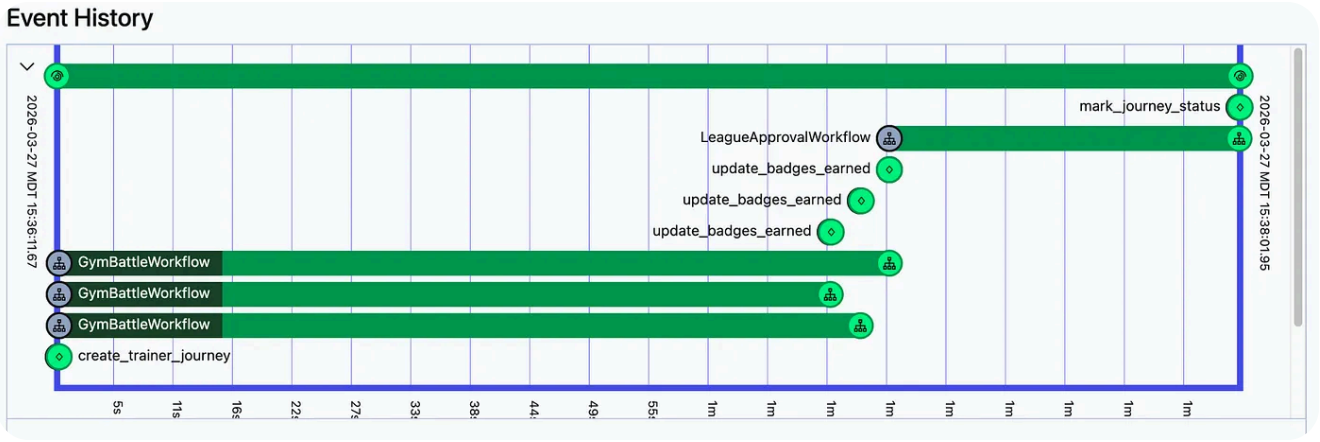
Tens of thousands of providers have gone through this system since we rolled it out, going live more accurately *and* faster than ever.

Beyond credentialing

We built Temporal into credentialing with the explicit goal that the patterns would be reusable.

As other teams saw what was possible with Temporal, they started building their own workflows on the same infrastructure. The Communications platform team uses Temporal to orchestrate sending emails, texts, and other outreach, coordinating delivery across channels in a way that's reliable and observable. Our Claims team uses Temporal to process and handle claims returned by payers, giving them better visibility into why claims are denied and tools to fix them at scale. Today, we have over a dozen worker pools running across teams including benefits, prescriptions, provider onboarding, billing, and more.

To make adoption easier, we invested in what we think of as the "paved road": reusable base classes for business processes and human-in-the-loop tasks, shared interceptors for metrics and tracing, a standardized way to spin up new worker infrastructure, and internal documentation. That way they can focus on the actual business logic and not worry about how to technically accomplish it. We even built an example app (a Pokemon gym challenge themed app) as a domain-neutral teaching tool so engineers could learn the patterns without needing any healthcare business context. It sounds silly, but having a reference implementation where "gym battles" map to parallel child workflows and "badge verification" maps to human-in-the-loop tasks turned out to be a really effective way to get people up to speed.



A view from the Temporal UI of the overall "Trainer Journey" workflow showing parallel workflows for each Gym and the follow-up approval process.

A view from the Temporal UI of a completed "Badge verification task" after a reviewer has signed off.

Why self-host

While Temporal offers Temporal Cloud to simplify running and maintaining it, we opted to self-host on our own infrastructure. Headway handles Protected Health Information (PHI), and we take the security of that data seriously. Self-hosting gives us full control over where data lives and how it's accessed. This obviously had more upfront work, but it's a trade off we prefer at Headway when it comes to protecting sensitive data.

Our infrastructure team did the heavy lifting to get Temporal production-ready, hardening the infrastructure, setting up per-team worker pools, building out observability, and tuning performance as usage scaled. That's a story worth telling on its own, and hopefully they'll write about it in a follow-up post.

What's next

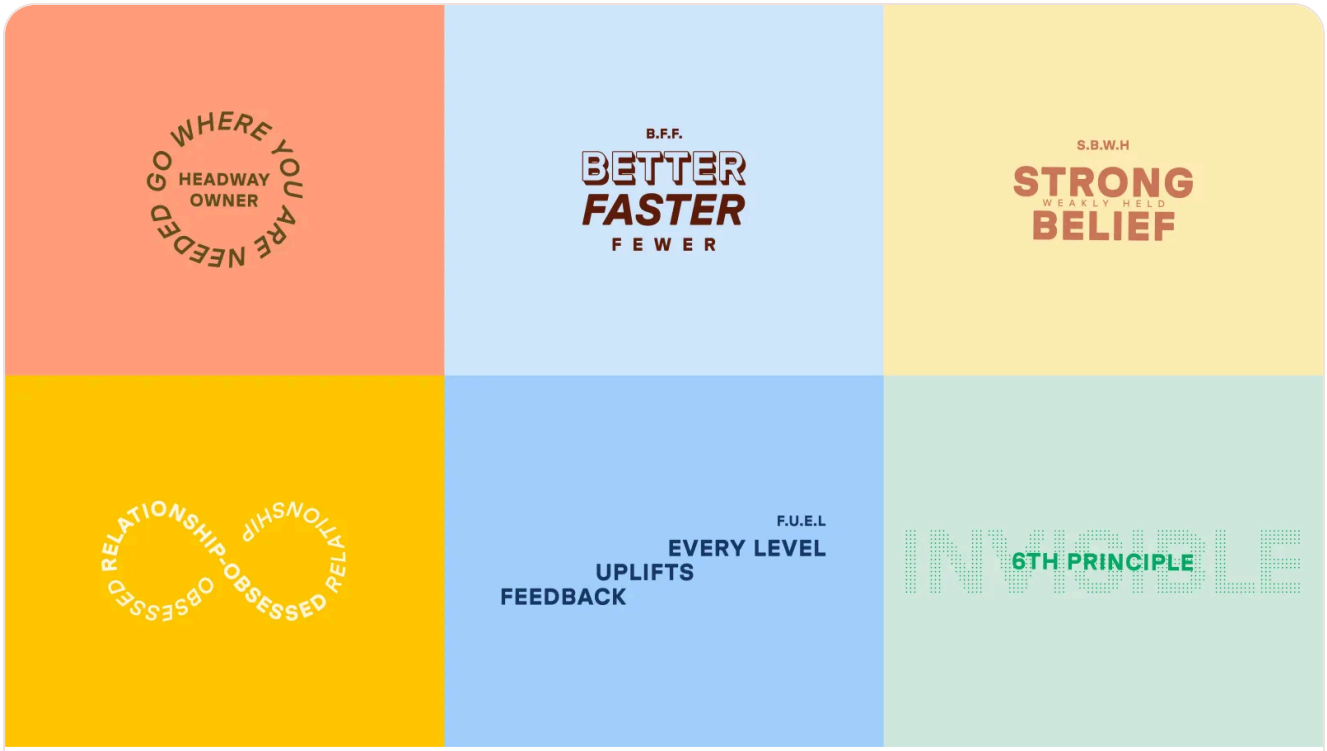
Temporal now runs workflows across more than a dozen teams at Headway, and we're still finding new problems it's well-suited for. If durable orchestration for complex, real-world processes sounds like interesting work and you care about making mental healthcare more accessible, come build with us.

SL

Stephen Leicht

Engineer

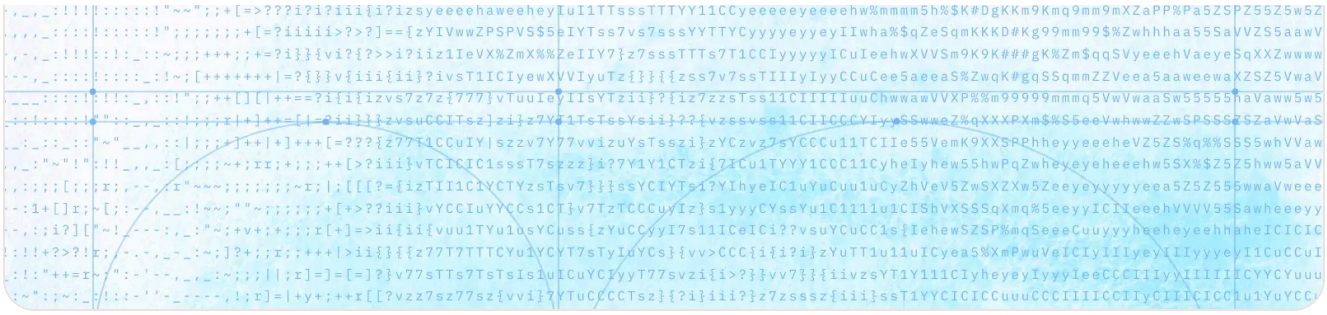
More from the blog



Our refounding cultural principles: what are they, and why did we create them?

COMPANY NEWS

Andrew Adams · Apr 22, 2025



Nearly 1 in 5 people seeking mental health care report thoughts of suicide

CLINICAL COMPANY NEWS

Dr. Nicole Benson · Aug 20, 2026

BUILD WITH US

Like how we think? Come build the future of mental health care.

We're hiring across engineering, clinical, and operations to make it easier for everyone to find care.

View open roles →

Headway

Find care

Therapists
Psychiatrists
By specialty
Online therapy
Therapy resources

Providers

Join Headway
Provider portal
Refer provider
Provider resources

Company

About us
Company blog
Press
Careers

Support

Help center
Contact us
For health plans
Sitemap
FAQs

If you're experiencing emotional distress, the resources below provide free and confidential support 24/7. **If this is an emergency, call 911**.

Suicide Prevention Lifeline
Call or text 988

Crisis Text Line
Text HOME to 741741

Terms of Service

Privacy Policy

Supplemental Payment Terms

HIPAA Notice of Privacy Practices

CA Privacy Policy for Business Customers

Washington State Consumer Health Data Privacy Policy

Biometric Data Policy

Accessibility

© 2026 Therapymatch, Inc.